Fermilab HEP Cloud Facility

Architecture and Design Overview

Version 1.0
August 10, 2016

CS-doc-5838

PREPARED BY:

Mine Altunay, Stuart C. Fuess, Gabriele Garzoglio, Burt Holzman, Robert Kennedy, Jim Kowalkowski, Steven Timm, Anthony Tiradani

Revision Log

Revision	Description	Effective Date
1.0	First Draft	08/29/2016
1.1	Added revision log, header, and modified footer, changed font	08/30/2016
1.2	Added section to discuss the user view of the Facility	09/16/2016

Table of Contents

1. Introduction

1.1 Purpose

This document provides the architectural description of the FNAL HEP Cloud Facility including a high-level breakdown of the key subsystems that compose the facility. This document will describe the major customer and stakeholder requirements that shape the system design. All detailed documentation will be derived from the architecture described here. The architecture and design of the components described in this document will be provided in supporting documents.

1.2 Project Description

Current Situation. Fermilab Scientific Computing supports several types of dedicated and shared resources (CPU, disk, hierarchical storage, including disk cache, tape, tape libraries), for both data intensive and compute intensive scientific work. This is limited, however, to resources provisioned by and hosted at Fermilab, or to remote resources made available through the Open Science Grid. The resources may be dedicated or shared and, in some cases, offered only at low priority such that their use may be pre-empted by higher priority demands on them. In order to reliably meet peak demands, Fermilab still must provision with the forecasted peak demand in mind, rather than the median or mean demand. This can be cost ineffective, since some resources may be underutilized during non-peak periods even with the resource sharing enabled by grids. This can also lower scientific productivity if the forecasted demand is too low, since there is a long lead-time to significantly increase current forms of local or remote resources.

Proposed Solution. The goal of the Fermilab HEP Cloud Facility Project is to extend the current Fermilab Facility to execute jobs submitted by the customers on disparate resources, including commercial and community clouds, grid federations, and HPC centers. The Fermilab HEP Cloud Facility will enable experiments to perform the full spectrum of computing tasks, including data-intensive simulation and reconstruction, at production scale irrespective of whether the resources are local, remote, or both. This will also allow Fermilab to provision scientific computing resources in a more efficient and cost-effective way, incorporating “elasticity”. This will enable the facility to respond to demand peaks without local overprovisioning, using a more cost-effective mix of local and remote resources.

1.3 Terminology

Compute Element (CE):	A service that provides access to a batch computing cluster, when the cluster is not directly accessible from the submission

	node. This service will authenticate and authorize jobs based on credentials passed along with the jobs. Sometimes referred to as a gatekeeper.
HTCondor-CE:	An HTCondor-CE is a specific implementation of a CE, utilizing HTCondor as its core technology. The external access protocol is condor-c.
High Performance Computing (HPC):	From TechTarget: "HPC is the use of parallel processing for running advanced application programs efficiently, reliably and quickly. The term applies especially to to systems that function above a teraflop or 10^{12} floating-point operations per second."
High Throughput Computing (HTC):	From Wikipedia: "HTC is a computer science term to describe the use of many computing resources over long periods of time to accomplish a computational task."
Pilot:	From Wikipedia: "A pilot job is a type of multilevel scheduling, in which a resource is acquired by an application so that the application can schedule work into that resource directly, rather than going through a local job scheduler, which might lead to queue waits for each work unit."
Virtual Organization (VO):	From Wikipedia: "A virtual organization is an organization involving detached and disseminated entities (from employees to entire enterprises) and requiring information technology to support their work and communication. Virtual organizations do not represent a firm's attribute but can be considered as a different organizational form."
Open Science Grid (OSG):	Open Science Grid (OSG): The main national grid infrastructure for HTC computing in the United States.
Worldwide LHC Computing Grid (WLCG):	The Worldwide LHC Computing Grid (WLCG) project is a global collaboration of more than 170 computing centres in 42 countries, linking up national and international grid infrastructures.
Stakeholder:	Stakeholders are persons or organizations that have an interest in the project process, output or outcome.
Customer:	Customers are stakeholders that pay for effort or other resources. They will also utilize the project output and will generate the target outcomes (benefits).

Partner:	Partners are stakeholders that collaborate in providing tools, software, or services to the project. Partners can influence the direction and priorities of the project.

1.4 HEP Cloud Customers

- Scientific Computing Division Office
- Compact Muon Solenoid (CMS) Experiment
- Open Science Grid (OSG)

1.5 HEP Cloud Partners

- Departments in Scientific Computing Division (SCD)
    - Scientific Data Processing Solutions (SDPS)
    - Scientific Distributed Computing Solutions (SDCS)
    - Experimental Computing Facilities (ECF)
- Departments in Core Computing Division (CCD)
    - Network and Communication Services (NCS)
- Compact Muon Solenoid (CMS) Experiment
- Open Science Grid (OSG)

1.6 Stakeholders

- FIFE Support
- FIFE experiments
- Compact Muon Solenoid (CMS) Experiment

2. Requirements and Constraints

2.1 Requirements

- Monitoring
    - Must provide audit logs for financial expenditures
    - Must provide audit logs for debugging purposes
    - Must provide audit logs detailing decisions made with regards to facility expansion and the inputs that lead to each decision.
    - Must provide detailed monitoring to the facility operations teams
    - Must provide high level monitoring to Fermilab management

    - May provide monitoring to customers of the facility
- Accounting
    - Must financially account for the use of the facility
    - Must provide accounting information for customer-submitted jobs executed on the facility
    - Must provide financial information for the facility use of computing credits administered on behalf of the customer.
- Security
    - Must create audit logs in a format consumable by Fermilab security team(s).
    - Must deliver security audit logs to security team(s) automatically and on demand.
    - Must provide Authentication and Authorization services for facility access
    - Must provide Authorization services for access to API or services (e.g. query API)
- Customer
    - Must execute authenticated and authorized customer-submitted work.
    - Must balance cost, performance, and deadlines on behalf of the customer for executing authorized customer-submitted work.
    - Must ensure the budget is not exceeded when executing customer-submitted work.
    - Must be capable of managing data transfers on behalf of the VO or user when the facility expands to external resources.
    - Must be capable of making automated decisions on behalf of the user/VO to determine whether to provision extra resources for particular customer-submitted work.
- Facility
    - Must extend the resources of the HEP Cloud facility as appropriate, based on demand, budget, and workflow types.
    - Must provide capability to expand to Cloud facilities.
    - Must provide capability to expand to HPC facilities.
    - Must provide capability to expand to Grid facilities.
    - Must manage resource life cycles on behalf of customers.
    - Must manage contractual and similar agreements related to the facility.
    - Must define a job manifest by which VOs and users can categorize the workflows submitted. The job manifest includes information such as, but not limited to, job completion deadlines and memory/cpu/disk requirements.
    - Must route customer-submitted work to specifically requested sets of resources
    - Must be able to create and maintain virtual machine images and container images on behalf of the facility and customer.

2.2 Constraints

It is important to note that these constraints are not a part of the overall architecture of the HEP Cloud Facility. These constraints apply specifically to current Fermilab instance of the architecture and inform specific examples and descriptions contained later in this document.

- The batch system used at Fermilab is HTCondor.
- glideinWMS is actively developed and used by both Fermilab and CMS. This software will be adapted and enhanced as necessary to satisfy the needs of the HEP Cloud Facility. Specifically, the glideinWMS factory will be used to provision HPC and Cloud resources.
- The Facility interfaces have already been defined as the external OSG Grid access interfaces and direct submission via JobSub.
- The local facility storage systems have already been selected and will not be changed due to this project.
- The upper limit of the expansion is bounded by the limitations of the individual experiment infrastructures, the capabilities of existing technologies used by HEP Cloud, and by subsystem constraints.

3. HEP Cloud Facility

3.1 HEP Cloud Overview

The current Fermilab Computing Facility offers a Grid-enabled[1] batch computing cluster to its customers. The Facility has allowed and will continue to allow opportunistic usage of spare computing cycles when there is lower demand from Facility customers. HEP Cloud is taking the next step by adding subsystems that enable the Facility to dynamically expand and contract the amount computing resources securely based on demand, budget, and other factors. This elasticity of resources is achieved by aggregating computing resources from HPC sites, cloud providers, and grid federations.

In addition to the computing offering, where required, the HEP Cloud Facility will manage the movement of data between local storage resources and remote storage resources, including HPC sites, cloud providers, and grid facilities.

[1] For a more detailed discussion on how a standard HTCondor batch computing system and a grid-enabled HTCondor batch computing system works, see Appendix A.

3.1.1 HEP Cloud Computing Overview

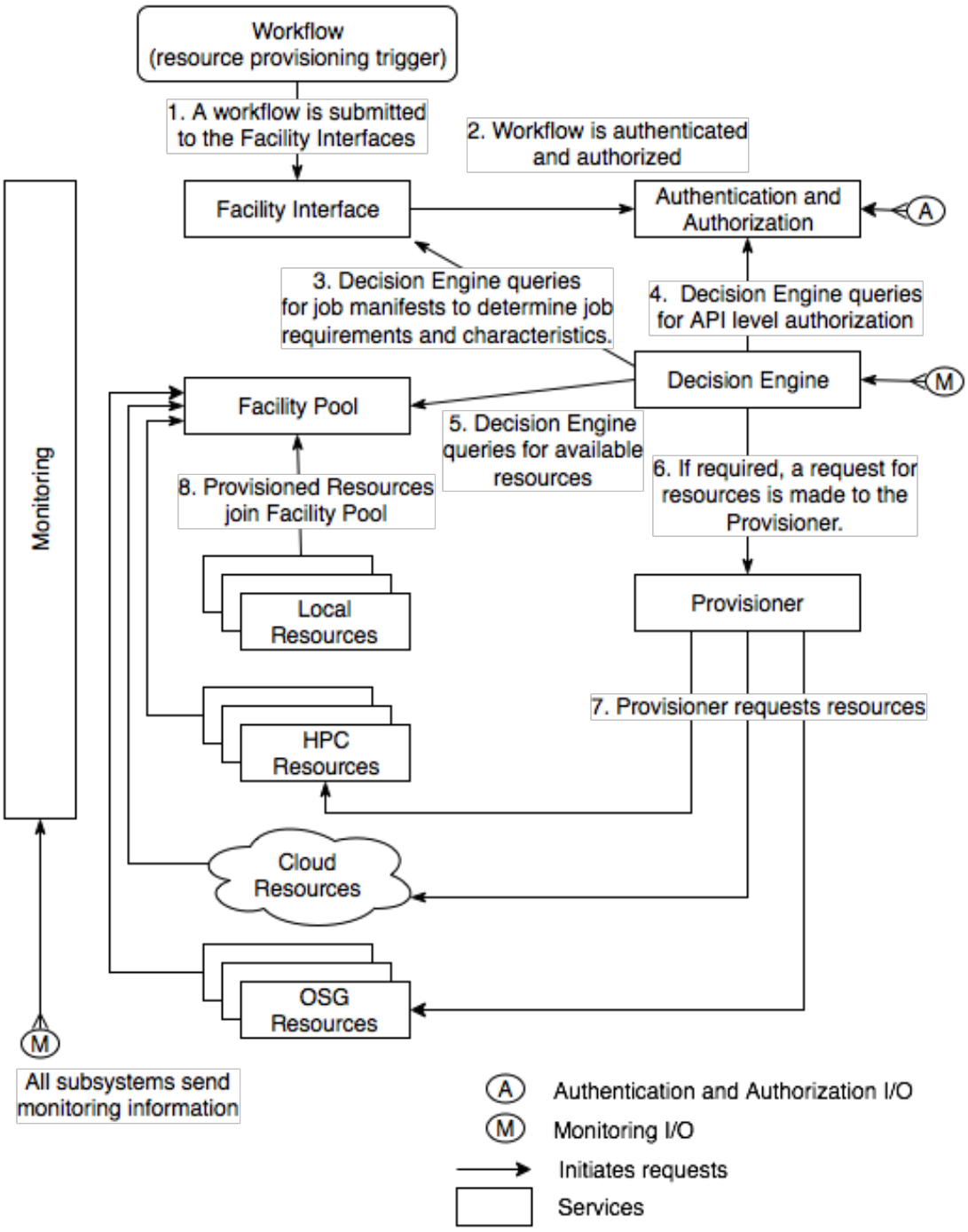

Figure 1: HEP Cloud Facility Provisioning

A VO submits workflows to the Facility as a series of jobs. As part of the job submission, the VO includes a manifest that describes the required resources and job behaviors. For example, the manifest may include memory constraints, network bandwidth requirements, or completion

deadlines. These jobs are authenticated and authorized using the VO submission credentials. This is the trigger that starts the chain of events to determine if an expansion of the Facility is required.

Once the jobs are authorized for Facility access, the Decision Engine queries the Facility interfaces for the jobs and their associated manifests. Based on the information obtained from the manifests, the Decision Engine determines suitable types of resources for the jobs. The Decision Engine queries the authorization service(s) to determine if the VO is authorized to run on the resource types considered (e.g. does this VO have an allocation at the HPC site, or is it authorized to run on the cloud?). The Decision Engine will then utilize monitoring information to determine if the resource providers are available, healthy, and have enough capacity. The Decision Engine will consider questions like "if the job is sent to the cloud, is there enough budget to complete the job?" At this point the Decision Engine decides if a resource request is warranted and, if warranted, will send the request to the provisioner. The Decision Engine will "tag" the jobs so that the Facility pool will only match the jobs with the requested resources.

Upon receiving a request from the Decision Engine, the Provisioner will request the resources from the provider and track the life cycle of the requested resources. The provisioned resources join the Facility pool and appear as standard worker nodes ready to execute the jobs that have been assigned to them.

Each of the subsystems or components in the Facility reports metrics and statistics to the monitoring subsystem and produce logs for auditing, debugging, and security analyses.

3.1.2 HEP Cloud Storage Overview

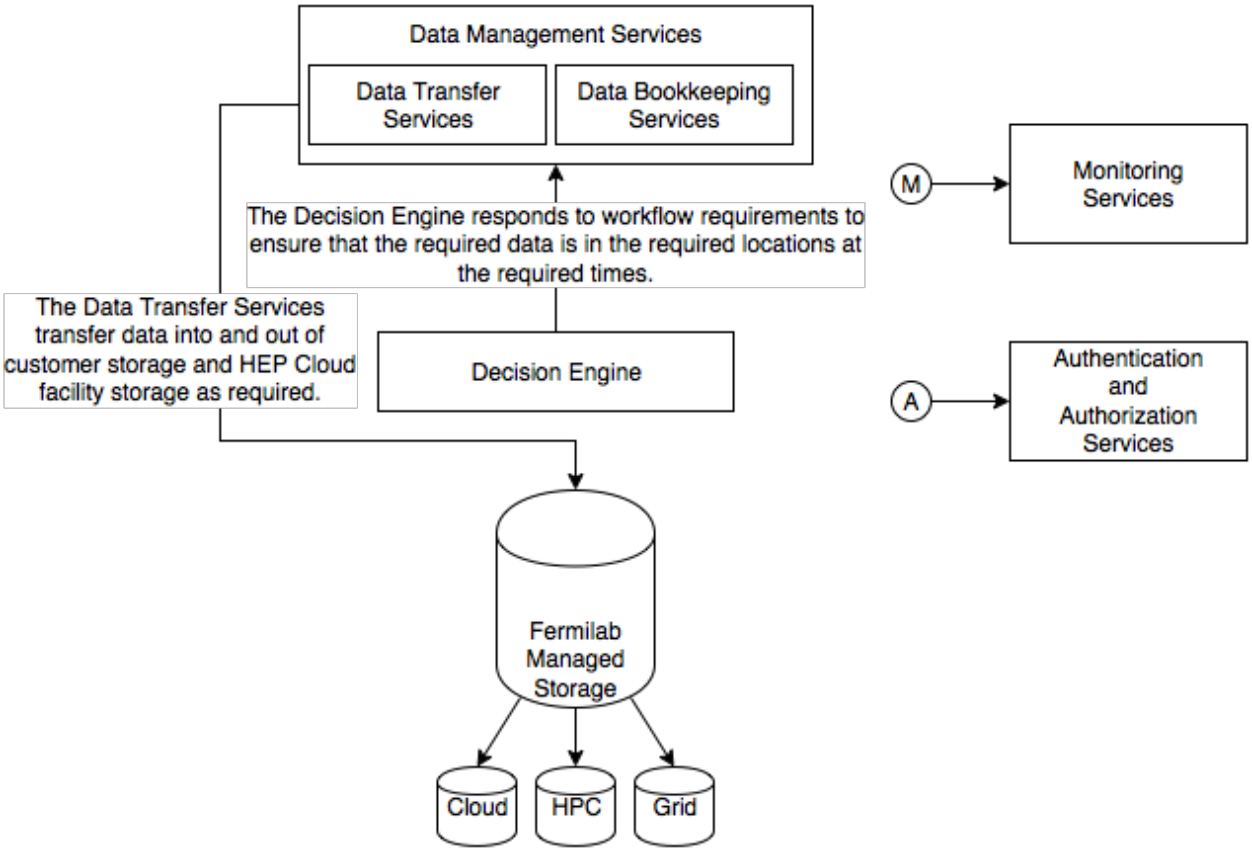

Figure 2: HEP Cloud Storage

The HEP Cloud Facility will transparently handle data transfers from local storage to a remote resource provider's storage subsystem. This can occur when a customer-submitted job specifies job locality requirements in the job manifest, or because built-up institutional knowledge of a particular resource provider suggests that a transfer is necessary. When a transfer condition is met, the Decision Engine will inform the Facility Data Management Services that the data needs to be transfered. These services are responsible for initiating the transfers, tracking the transfer status, and bookkeeping for the data. The Decision Engine is only responsible for making the request if appropriate.

As with all the other subsystems, the storage subsystems will request authentication and authorization from the Facility Authentication and Authorization services. The storage subsystems will also report monitoring information to the Facility monitoring services.

3.2 HEP Cloud Subsystems

The HEP Cloud Facility is broken down into subsystems listed in broad categories based on functionality. A subsystem is a service or group of services that can be isolated by functionality,

given to a subject area expert and owner, and developed separately from the entire HEP Cloud system. Each subsystem is briefly described and components of the subsystem contributed by Facility partners are identified. Partner contributed components are not developed or maintained by the Facility, but are integrated into the Facility to provide required functionality. Pointers to additional documentation for the subsystems are provided where available.

3.2.1 Facility Interfaces

The Facility interfaces are the entry points for work and/or data requests. This is where a user submits a job or requests that data is transferred into or out of the Facility. Each interface has a subsystem that takes computing resource requests and translates them into a job that is submitted to the job queue. The interfaces are required to submit their customer-submitted jobs to the Facility HTCondor Schedd. As mentioned in the constraints section, there are currently two computing interfaces pre-selected for the Fermilab Facility: the HTCondor-CE and JobSub. Both interfaces have at the their base level an HTCondor job queue.

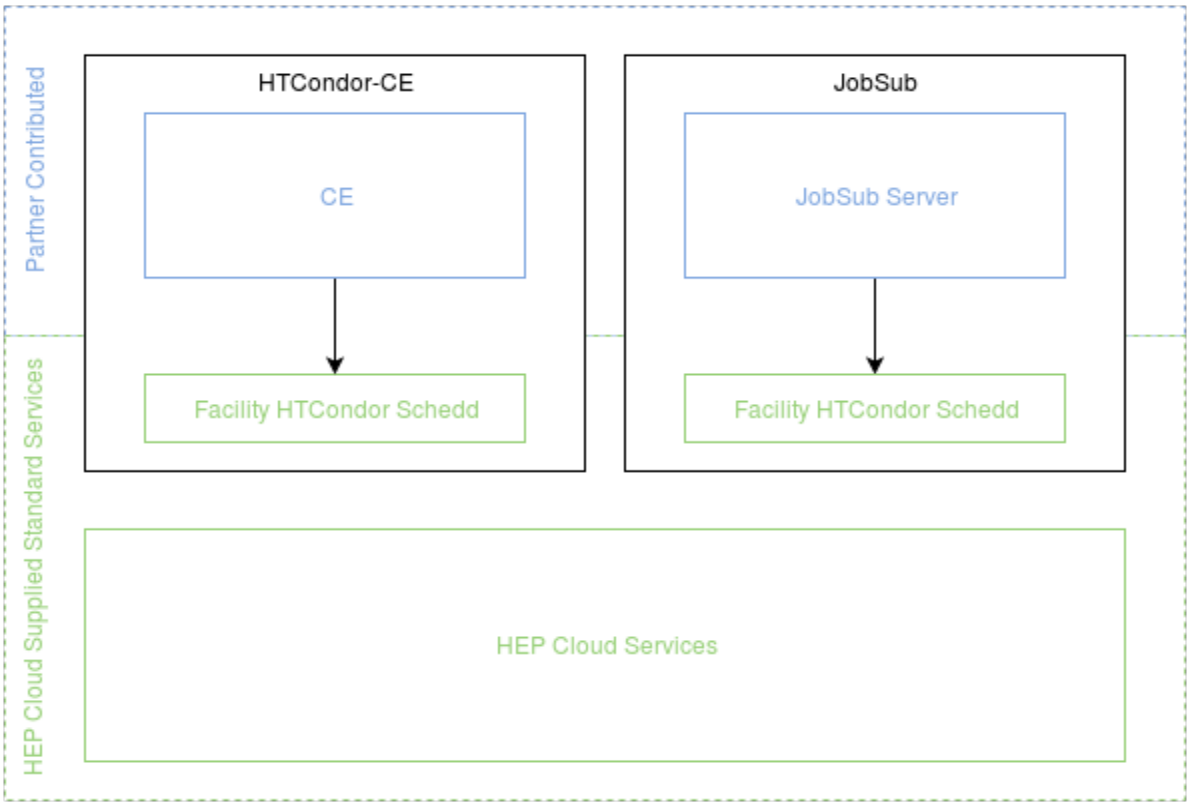

Figure 3: HEP Cloud Interfaces

The HTCondor-CE is an OSG-packaged HTCondor Schedd, HTCondor JobRouter, a set of configuration files, and supporting software that allows a remote VO to submit jobs to the Facility (see Appendix A). The JobSub server is a Fermilab-packaged set of software that allows Fermilab VOs to submit directly to the Facility. The main distinction, other than software differences, is that

the HTCondor-CE is expected to receive pilots from VO-operated pilot based workload management systems, whereas the JobSub server is expected to receive the actual VO jobs.

By allowing VO jobs to be submitted directly to the Facility, the Facility has firsthand knowledge of the user submitting the job and the characteristics of the job being submitted. This is especially important when considering the HPC resources. Some HPC providers have restrictions on the users allowed to run on their resources. Without firsthand knowledge, the Facility must trust the submitting VO to ensure that the HPC policies are enforced. Depending on the VO and HPC provider, this trust relationship may not be acceptable. Firsthand knowledge that comes with direct submission allows the Facility to enforce the HPC policies.

The two interfaces listed are the interfaces we currently anticipate using. However, we expect that additional interfaces will be added as HEP cloud evolves. The essential component from the Fermilab Facility perspective is the Facility HTCondor Schedd. The overall design of the Facility supports the case where another VO partners with Fermilab and HEP Cloud to contribute another interface to submit jobs to a Facility HTCondor Schedd.

An additional requirement for all HEP Cloud computing interfaces is that they must pass along or create a job manifest that will be considered by the Decision Engine. Because Fermilab is using HTCondor the manifest must be expressed somehow in the job ClassAd. The specifications for the job manifest content is in the Facility Interfaces subsystem documentation.

3.2.2 Authentication and Authorization

The Authentication and Authorization services verify the identity of the customer and determine whether the customer is allowed to perform an action. These functions are first triggered when a request of any kind reaches one of the Facility interfaces. That request must contain credentials with which to authenticate the entity making the request. Then the requests are either authorized or denied based on the authorization rules set up for that entity. Authorization steps continue throughout the system as different subsystems are queried or requests are made. The authorization component is a new development effort since this service does not currently exist.

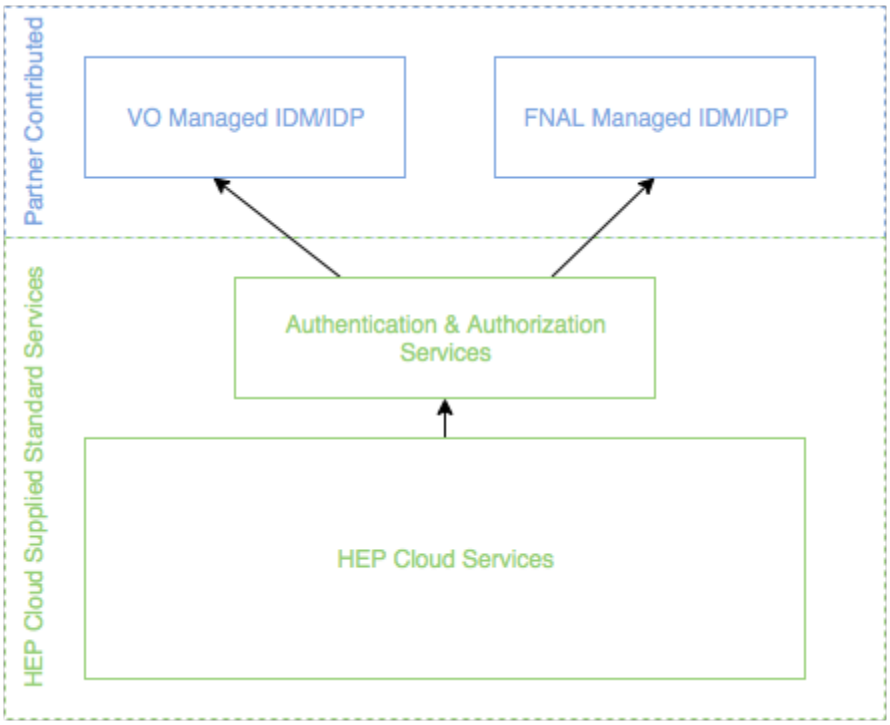

Figure 4: HEP Cloud Authentication and Authorization Services

Fermilab and HEP Cloud partners will operate and maintain some form of Identity Management (IDM) and Identity Provider (IDP) services. The HEP Cloud Authentication and Authorization services will interface with the partner contributed IDM and IDP services to establish identities within VOs. The HEP Cloud services will query the Authentication and Authorization services to ensure that only authorized users are able to submit jobs, request and receive external resources, make specific API calls to the resource providers, etc.

3.2.3 Decision Engine

HEP Cloud introduces to the Facility a new component currently called the Decision Engine that is responsible for determining if new resources are required. The Decision Engine is composed of a set of decision policies and the "engine" that ensures the decision policies are implemented. The policy implementation results in a request for resources from specific providers or potentially a decision to do nothing.

The Decision Engine policies consider specific workflow characteristics, resource capabilities, and/or resource costs. The Decision Engine policies use logic specific to each provider type to determine how many resources to request. For example, when making a request from a cloud provider, an exact number of virtual VMs may be included in the request, whereas a request to a grid federation may indicate a desired number of "idle" resources. Idle resources are resource requests pending execution. The Decision Engine policies make sure that the requests considered are authorized for access. For example, FIFE VO "A" makes a request to run a workflow on the Facility. It indicates that it would like to run on the cloud. Even before making any queries

regarding budget availability, resource availability, etc, the Decision Engine policies will first query the Authorization service to determine if VO "A" is allowed to make cloud requests. Only if they are authorized will the Decision Engine policies continue to determine if cloud resources are to be provisioned. Additionally, the Decision Engine policies take extra steps to ensure that workflows run on the resources that were provisioned for those workflows.

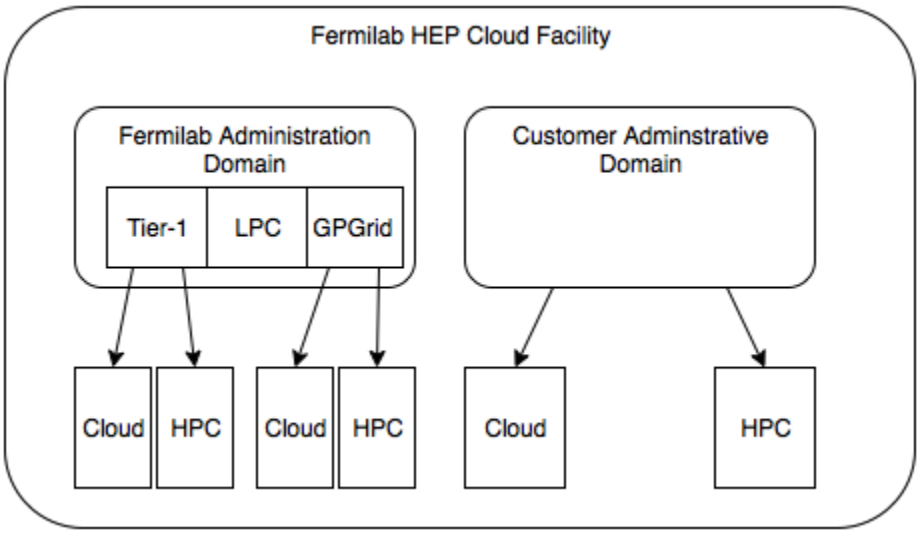

Figure 5: HEP Cloud Resources

The Fermilab HEP Cloud Facility has a fixed amount of computing credits available to execute jobs. These may include cloud credits, HPC allocations, as well as the local computing "credits". These different credits can be viewed conceptually as "currency" specific to different administrative domains (see Figure 5). The Facility customers have the option of bringing their own credits and requesting that the Facility use those credits to execute their jobs. The Decision Engine policies are responsible for ensuring that the customer administered credits are managed according to the desires and directives of the customer, while the Fermilab administered credits are managed according to the directives of Fermilab management.

3.2.4 Provisioner

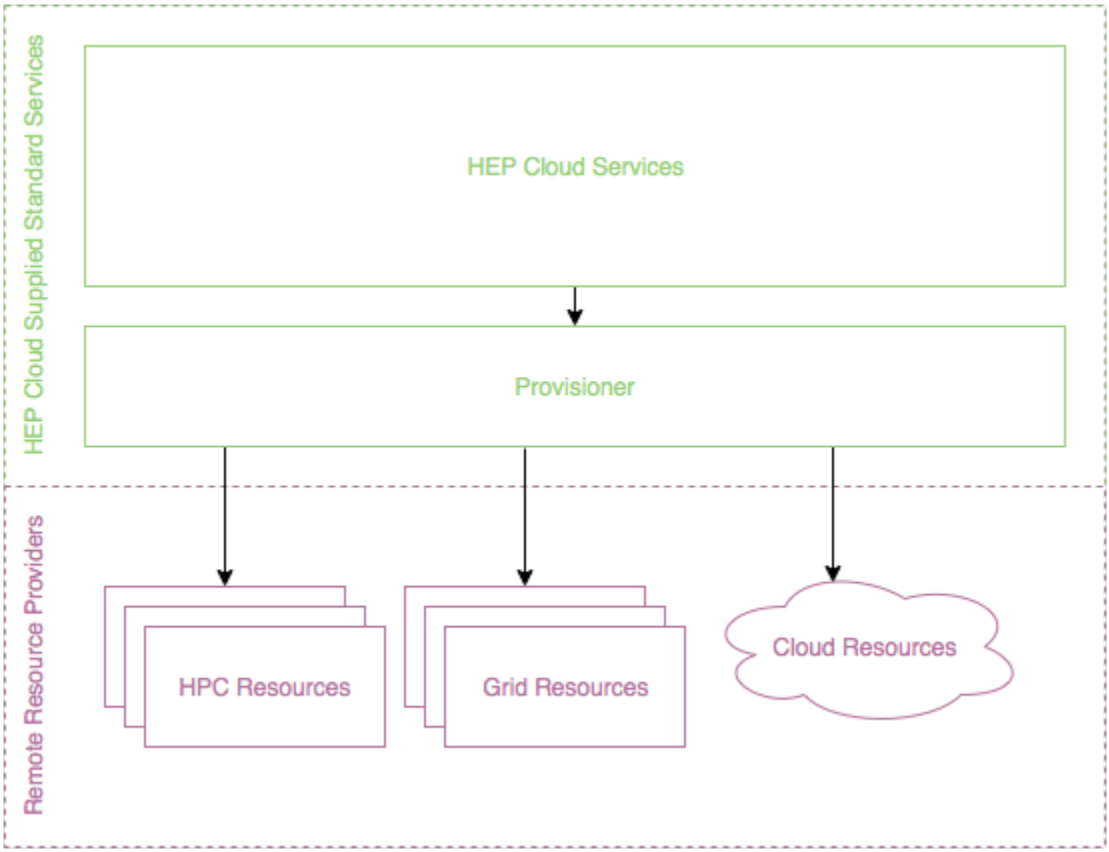

Figure 6: HEP Cloud Provisioner

The Provisioner is responsible for making resource requests to the individual providers whether they be cloud, HPC, or Grid providers. The provisioner only provisions resources by request from the Decision Engine policies. It will ensure that all requests are authorized and log all pertinent details about when resources are provisioned, how many, for whom, and for how long. This provides an audit trail for security purposes as well as providing information about the value and efficiency that the Facility is providing its customers.

The glideinWMS Factory[2] was chosen as the Fermilab HEP Cloud Facility provisioner because it is actively used, supported, and developed by Fermilab, the OSG, and CMS. From a support standpoint, there is a built-up expertise with the system that would be difficult to replace. From a functionality standpoint, the required functionality of provisioning HPC and Cloud resources is already built into the glideinWMS factory. This constraint applies to Fermilab, other facilities could use their own provisioner if desired.

[2] For further details on the glideinWMS factory see the glideinWMS project documentation. (http://glideinwms.fnal.gov/doc.prd/index.html).

3.2.5 Monitoring

The HEP Cloud Monitoring subsystem is responsible for checking that all the subsystems and components that make up the subsystems are functioning within acceptable parameters. The parameters range from basic functionality such as, "is the service running?", to more variable questions like, "is the Decision Engine making the proper decisions?". In some cases there may be automated corrective action that can and must be taken, in others, the monitoring subsystem simply needs to display the information for human consumption. With such a wide range of information gathering requirements and response requirements, it is anticipated that the Monitoring subsystem will be composed of many different components.

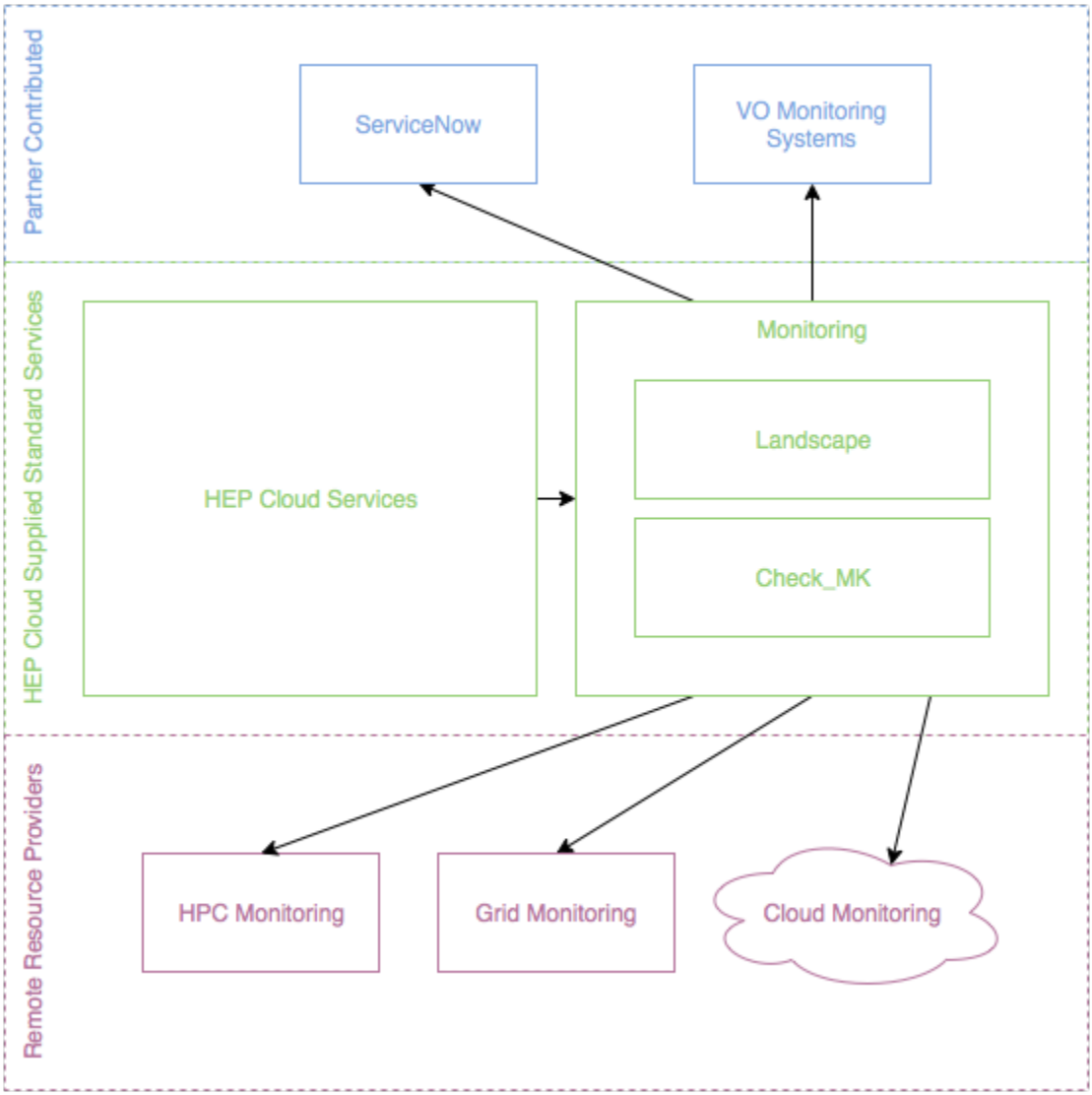

Figure 7: HEP Cloud Monitoring

The Fermilab HEP Cloud Facility monitoring subsystem re-uses many of the monitoring components already in use at Fermilab. Every day operations relies on the Landscape[3] project to provide trend analysis plots of system activity as well as instantaneous snapshots. This kind of monitoring assists operators in assessing overall behavior and health of the Facility.

Check_MK is used for monitoring the instantaneous state of the system. The checks run periodically and can be configured to perform a range of actions based on the check results. This is the system that will trigger a page to the system administrators if necessary.

ServiceNow is the service that manages the ITIL processes and relationships between the various services operated at Fermilab. This includes ticketing and paging. HEP Cloud will be fully integrated into ServiceNow and besides the ticketing, HEP Cloud depends on ServiceNow to page system administrators.

HEP Cloud makes use of provider native monitoring where available. For example, AWS provides a full suite of tools, technologies, and logging capabilities. HEP Cloud makes use of lambda functions to trigger ServiceNow alarms, and uses CloudTrail to keep audit trails for monitoring and security purposes. Whenever new providers, whether cloud, HPC, or otherwise, are added to HEP Cloud, any available native monitoring will be evaluated and incorporated as necessary.

HEP Cloud implements custom checks that interface with one or more of the listed systems, but do not specifically reside in any one of them. These checks include budget burn rates, total remaining budget, monitoring activity on remote providers that is attributed to HEP Cloud managed accounts, etc.

The architecture for HEP Cloud allows for the wholesale replacement of any of the monitoring technologies listed here.

3.2.6 Data Management Services

When computing requests are made to the Facility, there may be data locality requirements expressed as well. If the Decision Engine policies determine that a particular workflow requesting resources from the Facility is authorized and is best suited for the cloud, for example, but there are also data locality requirements associated with the request, then the Facility must move the required data into cloud storage. The Data Management Services aim to provide automatic, seamless data transfer between segments of the Facility when authorized and required as well as exposing interfaces for VOs or users of the Facility to make data movement requests.

Facility customers who have local storage allocated to them may need to make transfer requests for their own purposes. For this reason, the Data Management Services are at the boundary of the Facility and act as Storage interfaces as well.

[3] See the Landscape documentation for more details (https://landscape.fnal.gov/).

3.2.7 On-Demand Services

On-demand services are services that are instantiated at a site or within the provider facility at the time of need and torn down when the need no longer exists. For example, when preparing for the demonstrators, the Fermilab HEP Cloud Facility identified squid caching as a required on-demand service that needs to be implemented in the cloud. This service was only instantiated when cloud activity was expected and torn down when cloud use was terminated. Some of these services will result from accumulated operational experience, however, it is possible that a customer may have a service they would like instantiated prior to running a workflow. The HEP Cloud Facility will provide a service or set of services that will orchestrate the instantiation and tear-down of these services.

It is worth noting that the squid services are only the simplest of on-demand services. HPC sites pose new and interesting challenges since they traditionally have not had to serve customers with large and distributed infrastructures. The HEP Cloud Facility must have provisions for integrating HPC-specific services that help satisfy HEP needs. Examples are using various edge services and internal tooling that the HPC sites are willing to expose.

3.2.8 Auditing

Every subsystem in the Facility must write out logs detailing information about events that are occurring, errors, decisions made, API calls, and other information. The auditing subsystem is intended to collect these logs and provide data regarding security incidents, metrics related to performance issues, and debugging information necessary to verify system behavior.

3.2.9 VM and Container Management

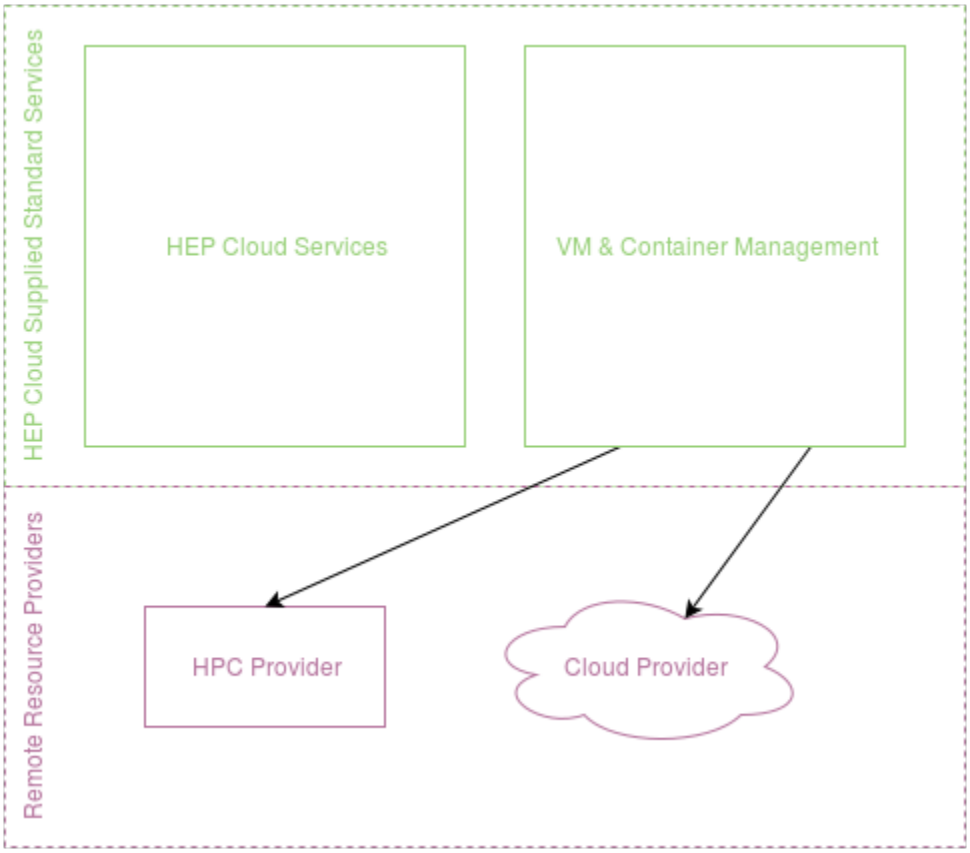

Figure 8: HEP Cloud Virtual Machine and Container Management

One goal of the HEP Cloud Facility is to provide a seamless computational experience regardless of where a particular workflow is run. This means that a cloud computing resource must look like a local computing resource as much as possible. To create the same environment on cloud providers, the HEP Cloud Facility creates and maintains custom Virtual Machine images with the same set of software installed that would ordinarily be installed on local machines. However, the Facility does anticipate that there may be unique job requirements or unique features that a particular provider may expose. In those cases, the Facility may create and maintain images that satisfy the unique requirements or features.

HPC providers are trending towards utilizing container technology, such as Docker, to allow their customers to create the specific environments that they need to execute their workflows. The HEP Cloud Facility may create and maintain container "images" that will provide a standard execution environment on HPC resources.

The virtual machine and container images created will follow the baseline, best practices, and security guidelines established by the Fermilab security team(s). This includes vetting and authorizing personnel who will be allowed to create and/or modify images on behalf of the Facility and Facility customers.

The subsystem(s) that handle the VM and Container management are supporting services that do not directly integrate into the Facility, but are necessary to maintain the Facility environments.

3.2.10 Accounting

Accounting services will provide the HEP Cloud Facility and its customers the ability to see which resource types were used, how many, and for how long. The accounting services will provide standard web based user interfaces that expose standard reports, graphs, and raw data access.

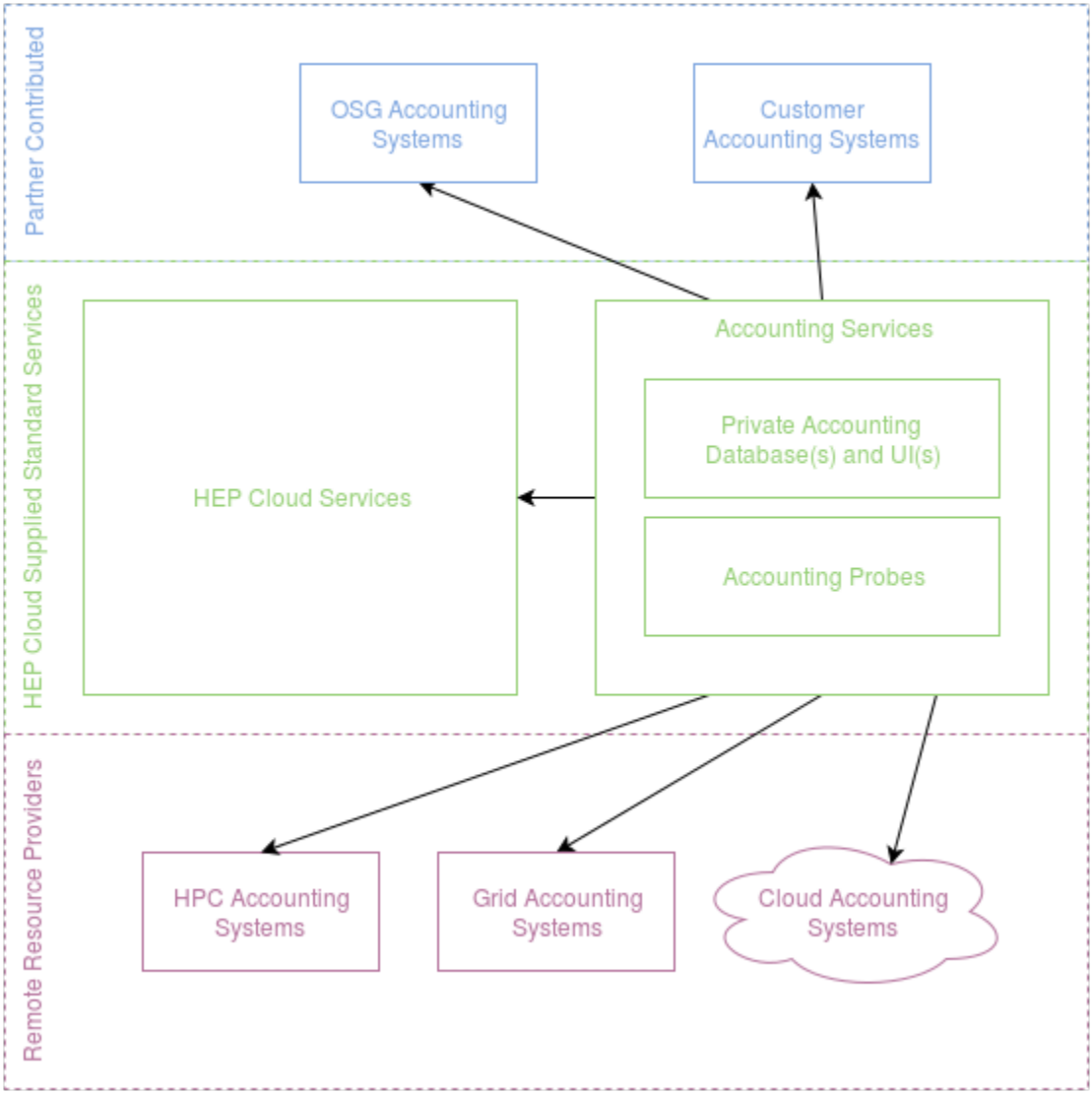

Figure 9: HEP Cloud Accounting

The accounting subsystem is fed by probes that run on the interfaces that report usage statistics from the Facility schedulers, by probes that collect usage information from the external resource providers using their native protocols, and by the resource job wrappers that pull in the customer-submitted jobs.

4. HEP Cloud External Relationships

4.1 HEP Cloud External System Interactions

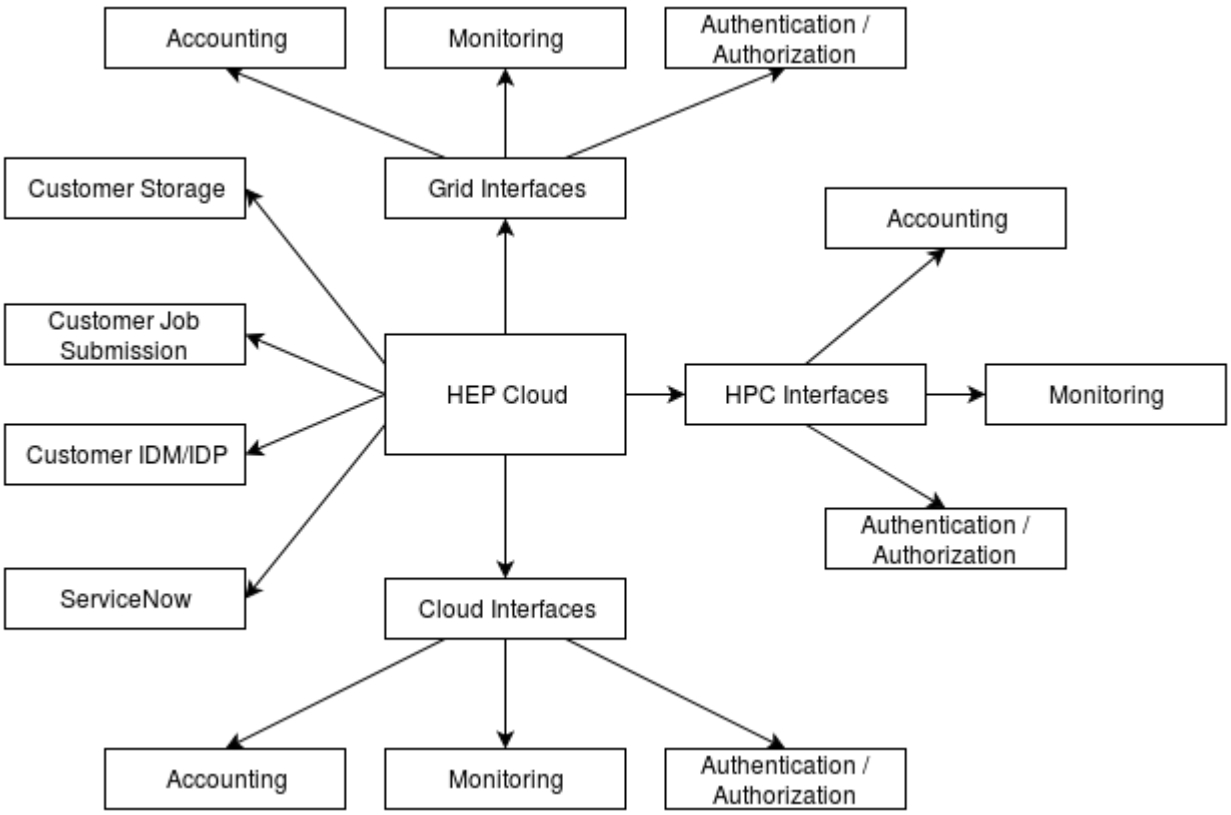

Figure 10: HEP Cloud External System Context

The HEP Cloud Facility interacts with many different external services (see Figure 10). The HEP Cloud Facility has its own monitoring and accounting systems; however, the Facility must also import the monitoring and accounting information from the different resource providers to ensure that all the different views of resource usage and performance are consistent. The Facility also must interact with customer infrastructure services such as the job submission service, or their storage systems.

4.2 HEP Cloud External Role Relationships

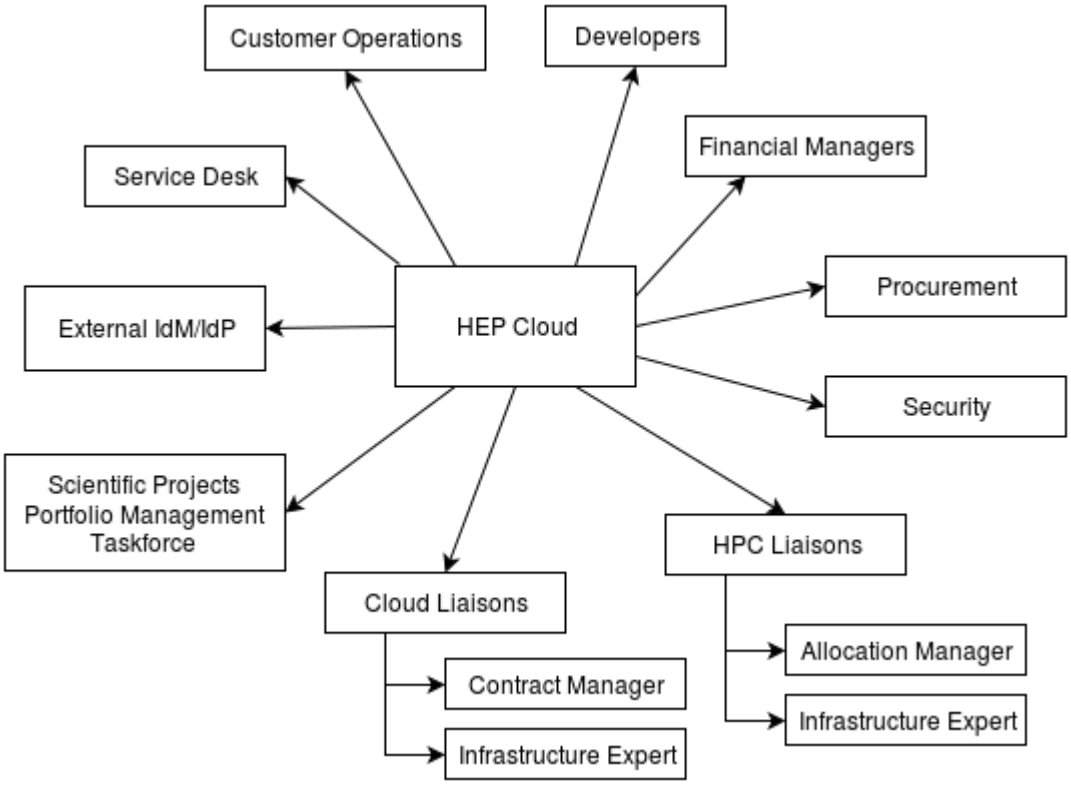

Figure 11: HEP Cloud External Role Context

Figure 11 represents the interactions of the HEP Cloud facilities with external organizations or people with specific roles. For example, the Scientific Projects Portfolio Management Taskforce will present experiment requirements and allocations to the Facility.

5. Example Deployment of the HEP Cloud Facility

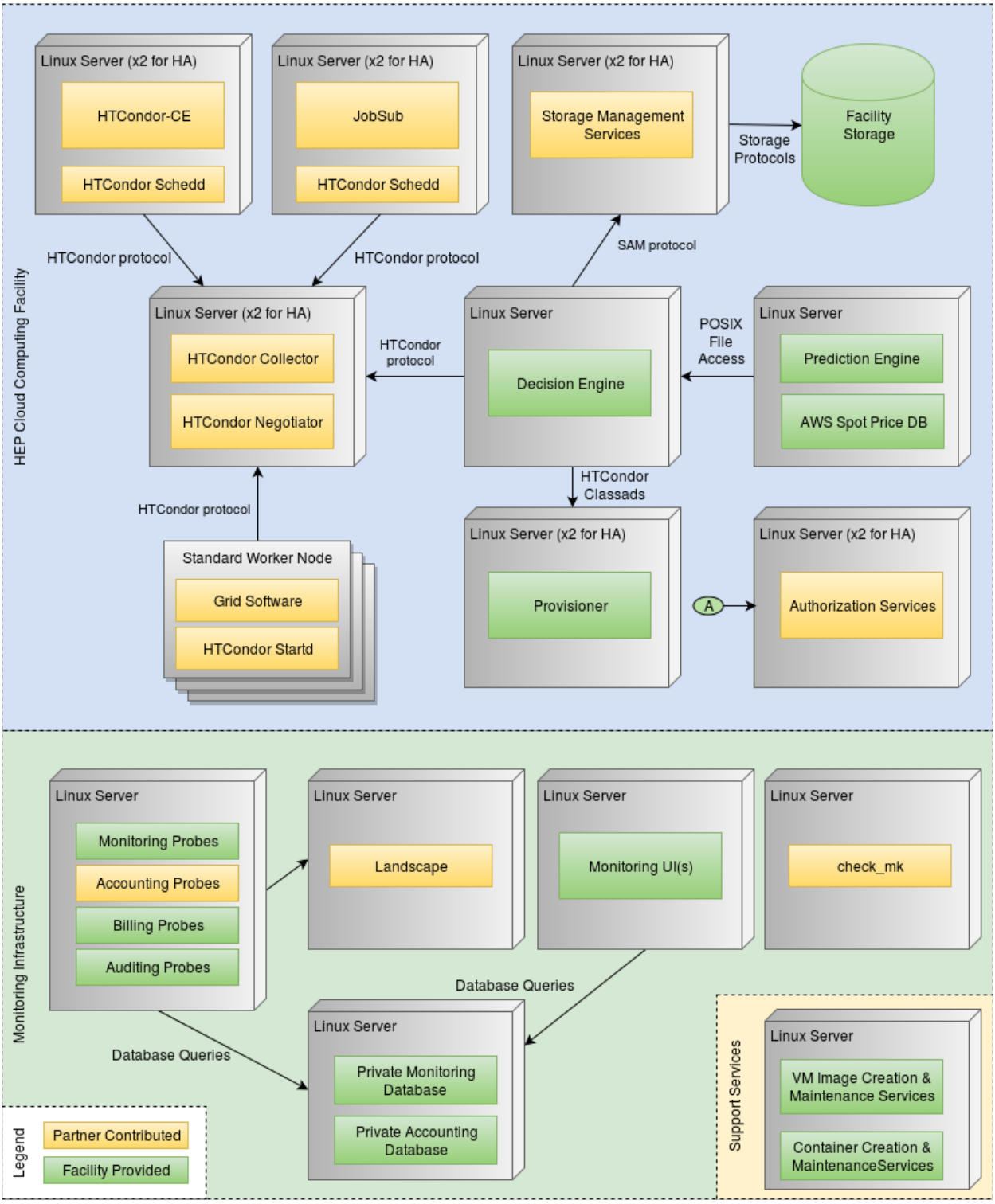

Figure 12: Example HEP Cloud Deployment Diagram at Fermilab

An example deployment of the Fermilab HEP Cloud Facility is shown in Figure 12. This is specific to how Fermilab might deploy the services that make up the Fermilab HEP Cloud Facility. The only services shown are services that the Fermilab Facility installs and operates. Services listed in the yellow boxes are partner contributed and services in the green boxes are Fermilab HEP Cloud services. The deployment is broken into two main areas. The monitoring infrastructure covers monitoring, accounting, and auditing services. The Fermilab HEP Cloud Computing Facility section covers the HTCondor batch system, Decision Engine, Provisioner, authorization services, and storage.

6. User View of HEP Cloud

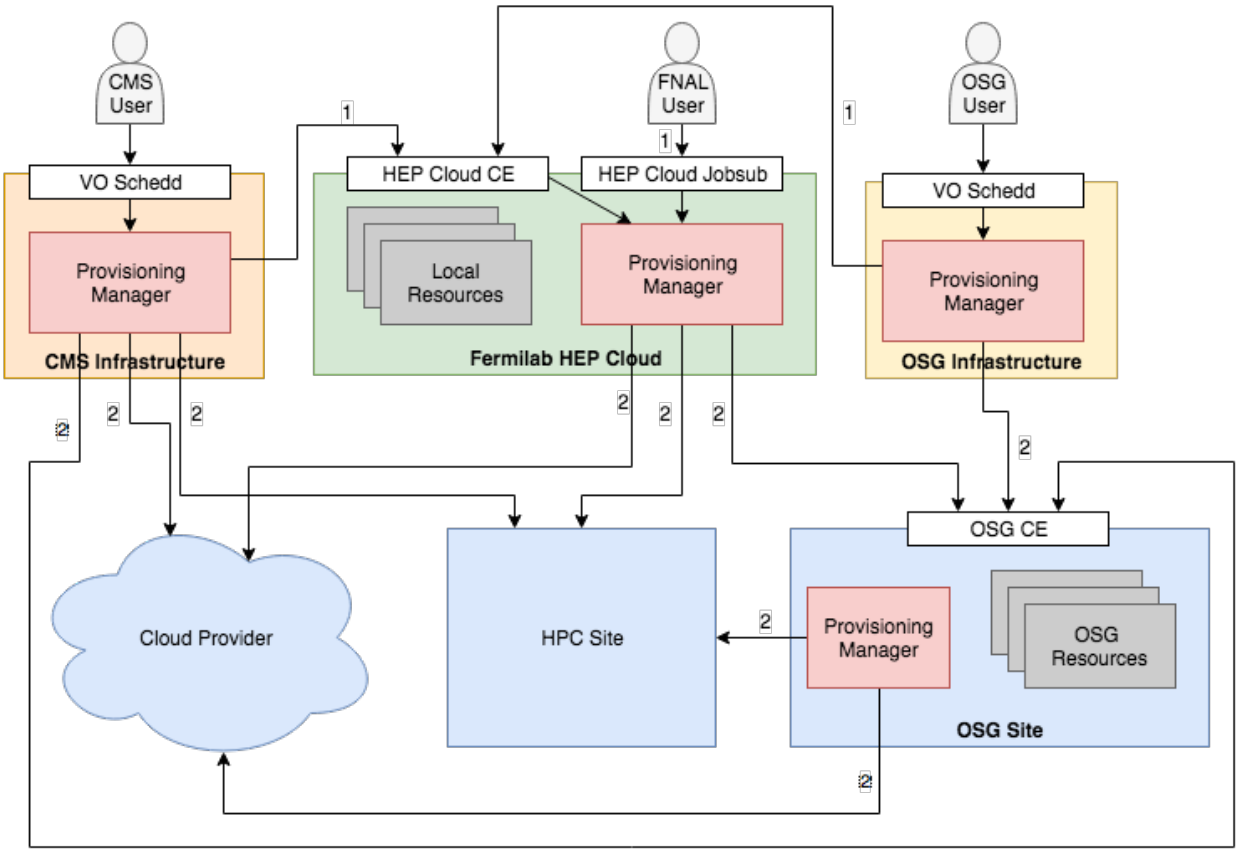

Figure 13: HEP Cloud User View

Three different user views are anticipated for the Fermilab HEP Cloud Facility. The first is internal Fermilab (FNAL) users. The FNAL user will use the Jobsub client to submit [4] jobs to the Jobsub interface. Internally, the Facility provisions [4] the appropriate resources for the FNAL user jobs.

[4] Figure 13: The number one indicates a user job submission path. The number 2 indicates a resource provisioning path.

The second user view is from the point of view of a fully supported external customer that has their own provisioning infrastructure. An example of this type of customer for the Fermilab HEP Cloud Facility is CMS. CMS users submit [4] to the CMS scheduler. The CMS provisioner requests resources from the Fermilab HEP Cloud Facility via the HEP Cloud CE. Again, internally, the Facility provisions [4] the appropriate resources for the CMS jobs.

The last user view is from the point of view of an opportunistic VO. The OSG VO is an opportunistic VO that has its own infrastructure similar to CMS. Like CMS, OSG makes resource requests via the HEP Cloud CE. Internally, the Facility provisions [4] the appropriate resources for the OSG jobs. The difference between a fully supported external customer and an opportunistic VO is mostly one of policy.

A key concept that can be gleaned from Figure 13 is that in each user view there are common components. All users submit jobs to a scheduler. For the example users described above, the scheduler used is HTCondor. Each user's jobs trigger a provisioning process by which resources are made available to run the jobs. Again, in the case of the example users, the underpinning technology used or proposed is HTCondor. In the end, the users all see an HTCondor pool which runs their jobs.

Appendix A

Standard HTCondor Batch Computing Cluster

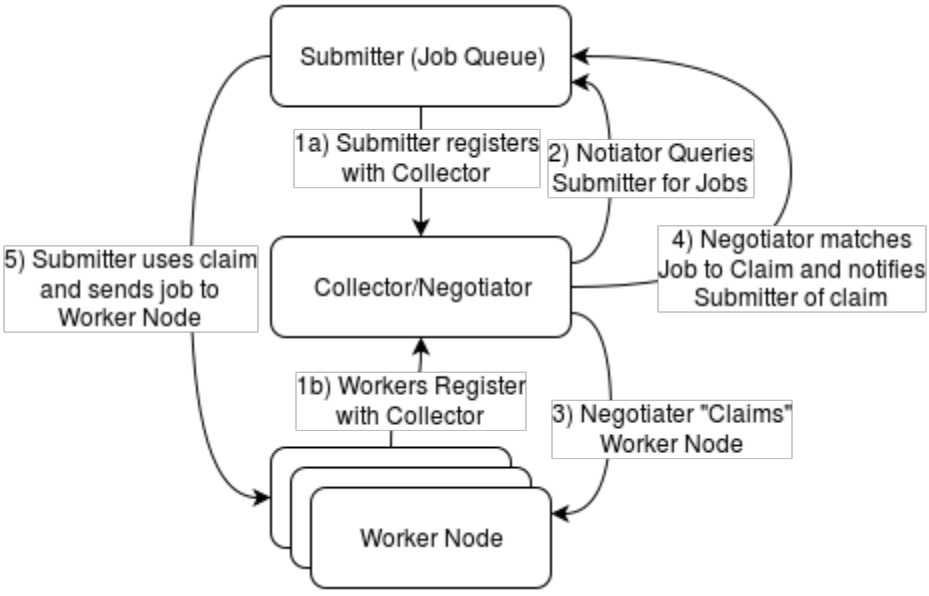

Figure 13: Standard HTCondor Batch Computing Cluster

A standard local HTCondor Computing cluster has one or more submitters, one collector, one negotiator, and one or more worker nodes. The submitters are the job queues where VOs submit their jobs. The worker nodes execute the jobs. The collector maintains the state of the cluster and provides necessary information to each of the components in the cluster. The negotiator matches jobs to available worker nodes.

The submitter(s) and worker node(s) register themselves with the collector. This enables the negotiator, submitter(s), and worker node(s) to discover each other during normal operation. The negotiator will discover all submitters and query each of them for any jobs that have been submitted. The negotiator also queries the collector for available worker nodes and matches the jobs to worker nodes. Once a match is determined, the negotiator "claims" a worker node on behalf of a job and notifies that worker node of the claim. The negotiator hands off the claim to the submitter. The submitter then contacts the worker node directly and passes the job to the worker node. The worker node then executes the job. For technical details please refer to the online documentation for the HTCondor project at the University of Wisconsin-Madison (https://research.cs.wisc.edu/htcondor/).

Grid Enabled HTCondor Batch Computing Cluster

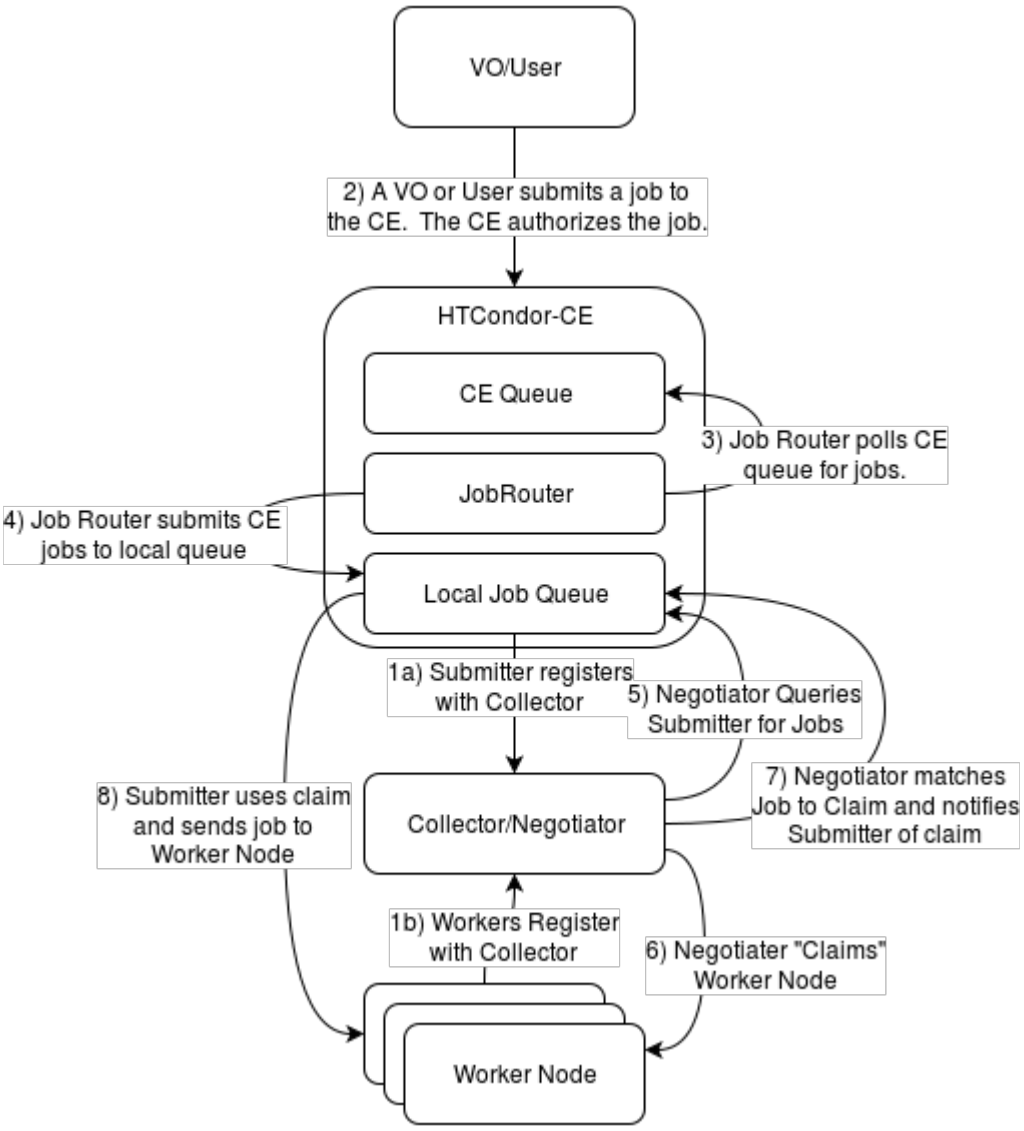

Figure 14: Grid Enabled HTCondor Batch Computing Cluster

A "gatekeeper" or CE is required to enable VOs to submit jobs to a cluster from remote locations. This creates a Grid enabled cluster that can participate in grids such as the OSG or the WLCG. The main difference between a local HTCondor computing cluster and the Grid enabled HTCondor cluster is the submitter. Instead of simply being a local job queue, the submitter is now a CE. Fermilab uses the HTCondor-CE in its Grid enabled HTCondor clusters. This CE utilizes a grid facing queue and a local facing queue. The grid facing queue is responsible for authorizing VOs for job submission. The CE uses a special component, the job router, to move or route jobs from the grid facing queue to the local facing queue, where HTCondor behaves as a local cluster.